FINAL PROJECT

I590 – PYTHON

TABLE OF CONTENTS

PROBLEM STATEMENT

The objective of the final project is to implement *k-means* algorithm for Wisconsin Breast Cancer data set using Python.

Breast cancer is a rising issue among women. A cancer's stage is a crucial factor in deciding what treatment options to recommend, and in determining the patient's prognosis. Today, in the United States, approximately one in eight women over their lifetime has a risk of developing breast cancer. An analysis of the most recent data has shown that the survival rate is 88% after 5 years of diagnosis and 80% after 10 years of diagnosis. With early detection and treatment, it is possible that this type of cancer will go into remission. In such a case, the worse fear of a cancer patient is the recurrence of the cancer.

In the final project we will address this issue by using *k-means clustering* to analyze cancer data and classify patients into two different groups, one with benign and the other one with malign cells. You will implement k-means clustering program in Python and test it on famous Wisconsin Breast Cancer Data.

DATA SET DESCRIPTION

Wolberg's breast cancer data can be found here. Samples arrive periodically as Dr. Wolberg reports his clinical cases. There are 11 columns (attributes) in this data set. The first column contains *Sample code number (SCN)*, which is the patient id number.

Column	Description	Name	Value
1	Sample code number	Scn	integer
2	Clump Thickness	A2	1 - 10
3	Uniformity of Cell Size	A3	1 - 10
4	Uniformity of Cell Shape	A4	1 - 10
5	Marginal Adhesion	A5	1 - 10
6	Single Epithelial Cell Size	A6	1 - 10
7	Bare Nuclei	A7	1 - 10
8	Bland Chromatin	A8	1 - 10
9	Normal Nucleoli	A9	1 - 10
10	Mitoses	A10	1 - 10
11	Class	Class	2 for benign, 4 for malignant

PHASE 1 (30%)

This project has been designed to be completed in three weeks. Each week you will deliver different phases of the project.

During the first week, you need to complete following data analysis tasks.

- Get yourself comfortable with k-means algorithm
- Download the data and load it in Python
- Impute missing values
- Compute data statistics
- Plot basic graphs

INSTRUCTIONS

a) Download breast cancer data from UCI machine learning repository.
Link:

https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data

b) Load dataset into Python.

Hint: you may use *pandas* library to load dataset. You may want to use option `na_values="?"` to replace missing values entered to column A7 as "?" with `NaN`.

Downloaded data is in file *"breast-cancer-wisconsin.data"* (without the names of the columns). The following statements read the data into data frame *df*, replace missing values with "?" and insert the names of the columns to the data frame:

```
col = ["Scn", "A2", "A3", "A4", "A5", "A6", "A7", "A8", "A9", "A10", "Class"]
df = pd.read_csv('breast-cancer-wisconsin.data',
                 na_values = '?', names = col)
```

c) Impute missing value to column A7. The missing values will be either "?" or NaN. You may replace missing values using techniques like mean, median, mode imputation or any other methods of your choice.

Example: If you want to impute missing values by 'mean' imputation method, you first need to compute the mean of column 'A7' without considering '?' data. Once you compute the mean, replace '?' value with computed mean value.

d) Find the mean, median, standard deviation and variance of each of the attributes A2 to A10. So you will have total of nine mean, median, variance and standard deviation values. Print all results rounded to 1 decimal place.

e) Plot histograms with 10 bins for attributes A2 to A10 (nine histograms).

Hint: A histogram is a kind of bar plot that gives a discretized display of value frequency. The data points are split into discrete, evenly spaced bins, and the number of data points in each bin is plotted. Use *Matplotlib* and "*hist*" method on the Series to plot a histogram.

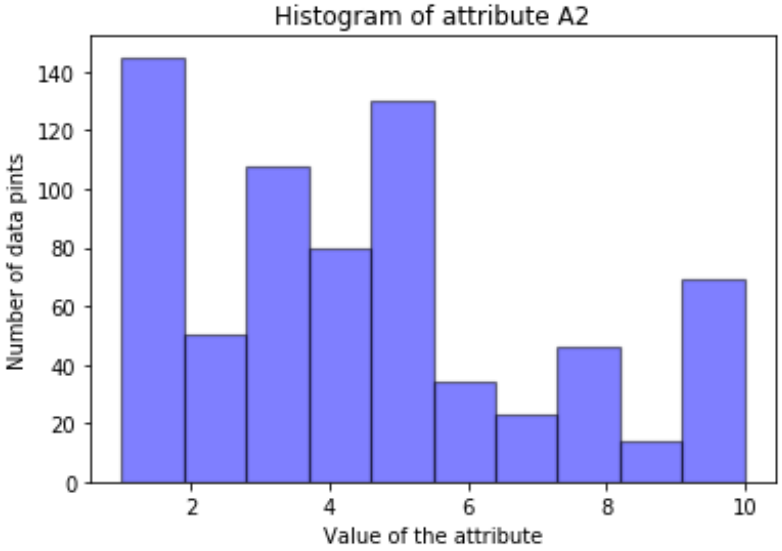
For example, if *s* is a *Series*, the following statement plots to subplot *sp* a histogram with 10 blue bins and opacity 0.5:

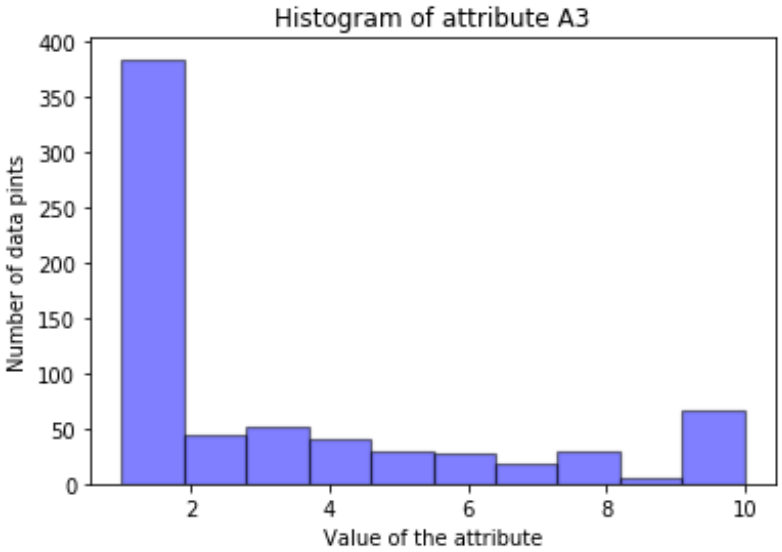
```
sp.hist(s, bins=10, color = "blue", alpha = 0.5)
```

SAMPLE OUTPUT

```
Attribute A2 ----------------
   Mean:               4.4
   Median:             4.0
   Variance:           7.9
   Standard Deviation: 2.8

Attribute A3 ----------------
   Mean:               3.1
   Median:             1.0
   Variance:           9.3
   Standard Deviation: 3.1
```





SUBMISSION GUIDELINES

- Submit your program for phase 1 and a PDF with all results. The results include printout with 9 tables (one for each attribute) and 9 histograms (again, one per attribute).
- Submit file “Breast-Cancer-Wisconsin.csv” if your program loads data from this file.
- You don’t have to write any formal project report in this week.
- No constraint on how many functions, classes and Python scripts you write to perform this task. However, you should have an executable Python script named ‘main.py’ that itself will be able to run your entire code by directly executing it.
- Provide ‘readme.txt’ file with explanation on how to execute your code and any additional information.
- Make sure your code is properly formatted, structured and commented.

PHASE 2 (40%)

You will implement k-means algorithm in the second phase. We suggest that you revise k-means clustering to make sure that you understand the algorithm before writing code.

In the Wisconsin Breast Cancer data set, 'Scn' is the first and 'Class' is the last column. We do not consider these two columns for k-means computation. However, you will need these two columns for printing results and, in phase 3, to calculate the errors.

Use the dataset with imputed missing values from phase 1 and use columns A2 to A10 for k-means computation.

INSTRUCTIONS

The steps in phase 2 include:

- Write code for 'Initial' step
- Write code for 'Assign' step
- Write code for 'Recompute' step

a) Write code for 'Initial' step

Randomly choose two points from data set as the initial centroids. Since you only consider column A2 to A10, an initial centroid is a nine dimensional vector. Give first centroid name μ_2 and second centroid name μ_4.

Example:

Let's say randomly selected data points are 6 and 246 as two initial centroids. So values of μ_2 and μ_4 are

$$\mu_2 = (8, 10, 10, 8, 7, 10, 9, 7, 1)\ ;\ \ \mu_4 = (5, 1, 1, 2, 2, 2, 3, 1, 1)$$

Note: You need to select two initial centroids randomly. So every time the code runs, two different initial centroids are selected. You may use 'numpy random' library to select random points from the data set.

b) Write code for 'Assign' step

For each one of 699 data points compute Euclidian distance from the two centroids. Use initial centroids in the first iteration. In the rest of the iterations use the centroids calculated in step *c* of the previous iteration.

At the end of this step, each data point will be assigned to one of the two clusters, which we call *predicted clusters*. Store the information on predicted clusters into a column *Predicted_Class*.

For each data point the value of *Predicted_Class* will be:

- *Predicted_Class = 2*, if the program calculates that the corresponding data point is in the cluster 2 that corresponds to centroid μ_2
- *Predicted_Class = 4*, if the program calculates that the corresponding data point is in the cluster 4 that corresponds to centroid μ_4

This step calculates two distances ror each data point. Assign a data point to cluster 2 (*Predicted_Class = 2*) if its distance from μ_2 is closer than its distance from μ_4; assign the data point to cluster 4 (*Predicted_Class = 4*) otherwise.

Example:

Let's take a data point at row number 375 and compute its distance *d* from both centroids.

data point 375 = (3, 1, 2, 1, 2, 1, 2, 1, 1)

$$d(375, \mu_2) = \sqrt{(3-8)^2 + (1-10)^2 + (2-10)^2 \ldots\ldots + (1-1)^2} = 20.25$$
$$d(375, \mu_4) = \sqrt{(3-5)^2 + (1-1)^2 + (2-1)^2 \ldots\ldots + (1-1)^2} = 2.83$$

Since $d(375, \mu_4) < d(375, \mu_2)$, data point at row number 375 will be assigned to cluster 4.

c) Write code for 'Recompute' step

So far, you have assigned each data point to one of the two clusters. Next, you will update the centroids.

Example:

Let's say after performing step *b*, you have 300 data points assigned to cluster 2 (i.e. *Predicted_Class* = 2), and remaining 399 data points assigned to cluster 4 (*Predicted_Class = 4*). Update μ_2 by computing the mean from cluster 2 data points and update μ_4 by computing the mean from cluster 4 data points.

d) Iterate steps *b* and *c* until any one of the following conditions is true:

1. Centroids (or clusters) don't change compared to the previous iteration.
2. Steps *b* and *c* iterated 50 times.

At the end step *d*, you will have final values of the centroids and predicted clusters. Print this result in console.

SAMPLE OUTPUT

```
Randomly selected row 148 for centroid mu_2.

Initial centroid mu_2:
A2     3.0
A3     1.0
A4     1.0
A5     3.0
A6     8.0
A7     1.0
A8     5.0
A9     8.0
A10    1.0
Name: 148, dtype: float64

Randomly selected row 190 for centroid mu_4.

Initial centroid mu_4:
A2     10.0
A3     10.0
A4     10.0
A5      8.0
A6      6.0
A7      8.0
A8      7.0
A9     10.0
A10     1.0
Name: 190, dtype: float64

Program ended after 4 iterations.

Final centroid mu_2:
A2     3.047210
A3     1.302575
A4     1.446352
A5     1.343348
A6     2.087983
A7     1.380001
A8     2.105150
A9     1.261803
A10    1.109442
dtype: float64

Final centroid mu_4:
A2     7.158798
A3     6.798283
A4     6.729614
A5     5.733906
A6     5.472103
A7     7.873966
A8     6.103004
A9     6.077253
```

```
A10    2.549356
dtype: float64

Final cluster assignment:

         Scn  Class  Predicted_Class
0    1000025      2                2
1    1002945      2                4
2    1015425      2                2
3    1016277      2                4
4    1017023      2                2
5    1017122      4                4
6    1018099      2                2
7    1018561      2                2
8    1033078      2                2
9    1033078      2                2
10   1035283      2                2
11   1036172      2                2
12   1041801      4                2
13   1043999      2                2
14   1044572      4                4
15   1047630      4                2
16   1048672      2                2
17   1049815      2                2
18   1050670      4                4
19   1050718      2                2
20   1054590      4                4
```

Note: Your output may be different than the above. Include the first 20 data points (rows) in your report.

This version of k-means algorithm may suffer from poor initialization. Therefore, you may see different answers or swapped cluster assignments. We recommend running program multiple times and submitting the best results.

SUBMISSION GUIDELINES

- Submit your program for phase 2 and a PDF with all results. The results should include the initial and final centroids and cluster assignment with first 20 data points.
- Submit file "Breast-Cancer-Wisconsin.csv" if your program loads data from this file.
- You don't have to write any formal project report in this week.
- No constraint on how many functions, classes and Python scripts you write to perform this task. However, you should have an executable Python script named 'main.py' that itself will be able to run your entire code by directly executing it.
- Provide 'readme.txt' file with explanation on how to execute your code and any additional information.
- Make sure your code is properly formatted, structured and commented.

PHASE 3 (30%)

Phase 2 program, which implements k-means algorithm, produces two clusters – one containing benign cells (predicted class = 2) and the other one that contains malign cells (predicted class = 4). But there are chances that a malign cell is clustered into a benign cluster and vice versa.

In phase 3 you will analyze the quality of the clustering. To check how well your clustering worked, you will calculate the error rate for your clusters. Assume that the column "Class" of the initial data set contains correct clustering of the data points.

INSTRUCTIONS

There are two parts in phase 3:

- Write a code to calculate the individual and total error rates of the predicted clusters.
- Prepare and submit final report

a) Write code to calculate the individual and total error rates of the predicted clusters

Your phase 3 program will calculate the error rates based on two arguments:

- The predicted clusters, calculated by your phase 2 program,
- The correct clusters, specified by the column "Class" of the initial data set.

Let's have a look at the example of the cluster assignment with first 20 data points, listed on page 8. Column "Class" represents the correct clusters and column "Predicted_Class" represents the clusters calculated by the k-means algorithm.

	Scn	Class	Predicted_Class
0	1000025	2	2
1	1002945	2	4
2	1015425	2	2
3	1016277	2	4
4	1017023	2	2
5	1017122	4	4
6	1018099	2	2
7	1018561	2	2
8	1033078	2	2
9	1033078	2	2
10	1035283	2	2
11	1036172	2	2
12	1041801	4	2
13	1043999	2	2
14	1044572	4	4
15	1047630	4	2
16	1048672	2	2
17	1049815	2	2
18	1050670	4	4

```
19   1050718      2                2
20   1054590      4                4
```

Marked data points represent the errors of the k-means clustering:

- Yellow data points are predicted as class 4 (malign cells), while the correct class is 2 (benign cells).
- Gray data points are predicted as class 2 (benign cells), while the correct class is 4 (malign cells).

Let's define the following notation:

error_24:	number of data points predicted as class 2, while the correct class is 4
error_42:	number of data points predicted as class 4, while the correct class is 2
error_all:	number of data points with predicted class not equal to correct class
pclass_2:	number of data points with predicted class equal to 2
pclass_4:	number of data points with predicted class equal to 4
class_all:	number of data points
error_B:	error rate for the benign cells
error_M:	error rate for the malign cells
error_T:	total error rate

Use the following formulae to calculate and print error rates for each cluster:

*error_B = (error_24 / pclass_2) * 100 %*

*error_M = (error_42 / pclass_4) * 100 %*

*error_T = (error_all) / (class_all)) * 100 %*

Total error rate more than 50% indicates that your program swapped the predicted clusters. Your program has to detect this situation, swap the predicted clusters by replacing 2 with 4, and 4 with 2 in column "Predicted_Class", and recalculate the error rates.

b) Prepare final report that incorporates all the results and your conclusions for phases 1 to 3.

SAMPLE OUTPUT

This is the output in case the clasters are swapped and the program swapped the predicted class.

```
Total errors:                     96.0 %
Clusters are swapped!
Swapping Predicted_Class

Data points in Predicted Class 2:  464
Data points in Predicted Class 4:  235

Error data points, Predicted Class 2:

         Scn  Class  Predicted_Class
12   1041801      4                2
```

```
15   1047630      4               2
50   1108370      4               2
51   1108449      4               2
57   1113038      4               2
59   1113906      4               2
63   1116132      4               2
65   1116998      4               2
101  1167439      4               2
103  1168359      4               2
105  1169049      4               2
222  1226012      4               2
273   428903      4               2
348   832226      4               2
356   859164      4               2
455  1246562      4               2
489  1084139      4               2

Error data points, Predicted Class 4:

         Scn  Class  Predicted_Class
1    1002945      2               4
3    1016277      2               4
40   1096800      2               4
196  1213375      2               4
252  1017023      2               4
259   242970      2               4
296   616240      2               4
315   704168      2               4
319   721482      2               4
352   846832      2               4
434  1293439      2               4

Number of all data points:         699

Number of error data points:       28

Error rate for class 2:            3.7 %
Error rate for class 4:            4.7 %
Total error rate:                  4.0 %
```

SUBMISSION GUIDELINES

- Prepare and submit a PDF with final report that includes:
  - Project statement
  - Short description of phase 1, 2 and 3 programs (algorithm, description of input data, structure of the programs and description of results)
  - Phase 1, 2 and 3 results
  - Conclusion
- Submit phase 1, 2 and 3 programs together with any data files that may be needed to run your programs.
- Provide ‘readme.txt’ file that provides information about how to execute your code.